Ongoing work on NSF OCI-1127228 at UNH InterOperability Laboratory

Robert D. Russell <rdr@unh.edu>

InterOperability Laboratory &
Computer Science Department
University of New Hampshire
Durham, New Hampshire 03824-3591, USA

UNH IOL Participants

- Bob Russell
  - P.I.
- Patrick MacArthur
  - UNH CS undergraduate student 2011-2012 AY
  - UNH CS Ph.D. graduate student 2012-2013 AY
  - UNH CS Ph.D. graduate student, NSF GRFP 2013-2014 AY
- Tim Carlin
  - UNH CS M.S. graduate student 2012 CY
- Qian Liu
  - UNH CS Ph.D. graduate student 2013-2014 AY

Close Collaborators

- University of Virginia
  - Malathi Veeraraghavan, P.I.
  - Zhengyang Liu, Ph.D. graduate student
- National Center for Atmospheric Research
  - John Dennis, P.I.

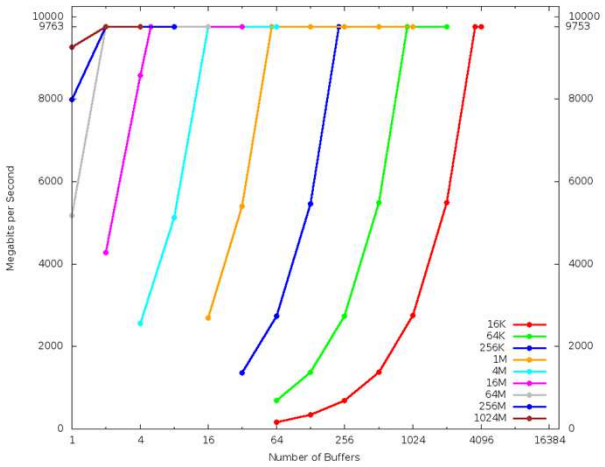
Overview

- Project goals
  - identify causes of variation in application-level performance
    - within the data center
    - between data centers
  - Upgrade the identified weakest components
  - Transfer the upgraded applications and tools to the broad scientific community

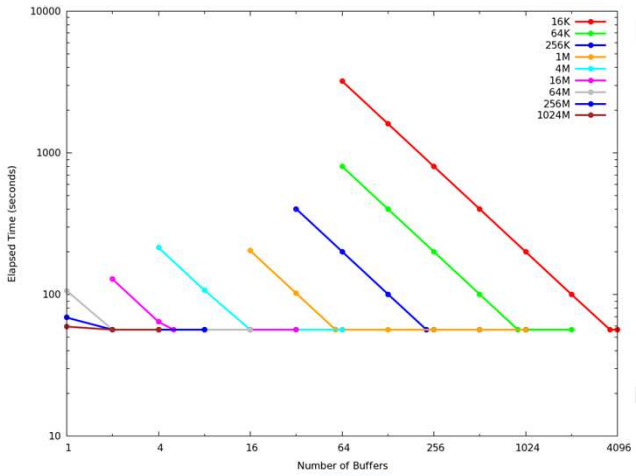
RDMA driver for GridFTP

- Key value proposition
  - No Operating System involvement in I/O transfers
  - Control and data move directly between user and NIC
- GridFTP driver – Tim Carlin
  - Performance depends on number of outstanding messages, message size, and RTT
  - cs.unh.edu/thesis-and-documentation-topics/
- Conclusions
  - GridFTP default buffer size, 256 Kibibytes, much too small
    - 256 Mebibytes is smallest at which RDMA outperforms TCP
  - GridFTP fixed double buffering bad for RDMA
    - Optimal number of buffers depends on buffer size and RTT

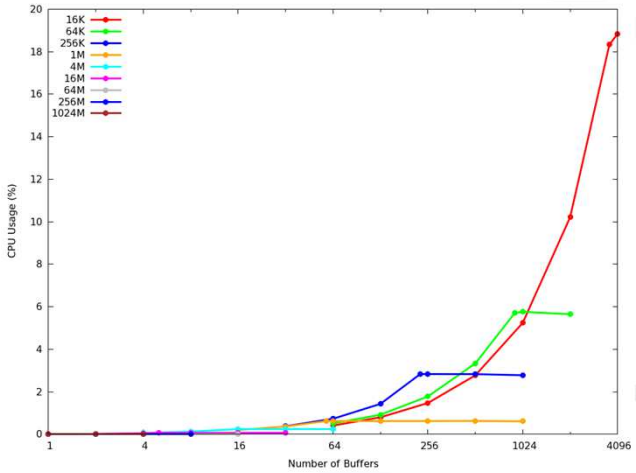
RDMA_WRITE throughput at 48 ms RTT



Elapsed time for 64 GibiBytes



Total client CPU for 64 GibiBytes



Predicting “best” number of buffers

- MNB = mimimum “best” number of buffers
  - MNB = ceiling((RTT * 1.25 + MTS + ATS) / MTS)
- RTT = round trip time (measured on connection)
- MTS = message transfer size
  - includes all frames and all frame overhead for 1 message on the wire
- ATS = RDMA acknowledgment transfer size

Observed vs Predicted N buffers

message size	observed		predicted nbuffers	error
	Mbps	nbuffers		
16 Kibi	9753	3638	3638	0
64 Kibi	9760	913	911	-2
256 Kibi	9762	229	229	0
1 Mibi	9762	58	58	0
4 Mibi	9762	16	16	0
16 Mibi	9763	5	5	0
64 Mibi	9763	2	2	0
256 Mibi	9763	2	2	0
1 Gibi	9763	2	2	0

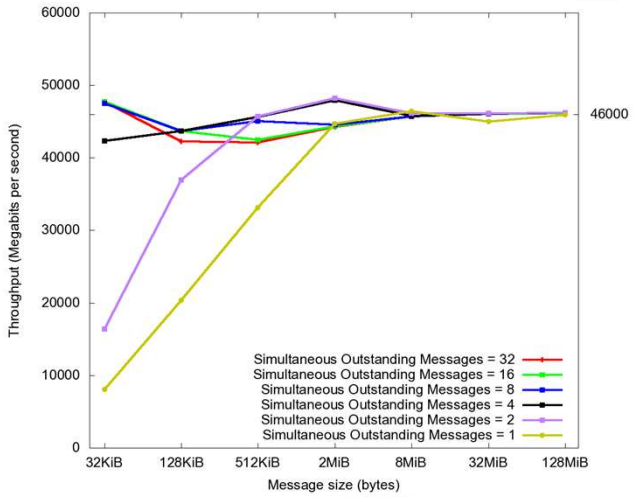
Related RDMA Technology Transfer

- Course Module on Data Center Networking
  - Introduction to RDMA programming
  - www.cs.unh.edu/~rdr/rdma-intro-module.ppt
- High-level interface to RDMA – UNH-EXS
  - Based on published Open Group Standard ES-API
  - Intended to make RDMA programming more accessible
  - Open-source software for new projects not needing MPI
  - Runs on InfiniBand, RoCE, iWARP (at all speeds)
  - Software and documentation at:
    iol.unh.edu / services / research / unh-exs

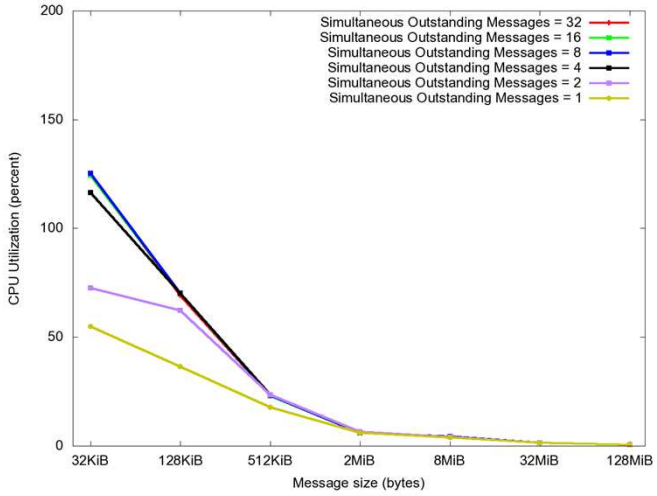
UNH EXS Developments

- Provides both
  - SOCK_SEQPACKET – reliable datagram service
  - SOCK_STREAM – reliable byte-stream service
- Announced and presented at OFA user-day 19-apr-13
  - Current release level 1.3.1
- Testing with industry partner
- New dynamic algorithm to handle SOCK_STREAMs
  - Preserves zero-copy transfers as much as possible
  - Reduces latency
  - Preserves high bandwidth utilization

EXS Blast Throughput over FDR



EXS Blast CPU Usage over FDR



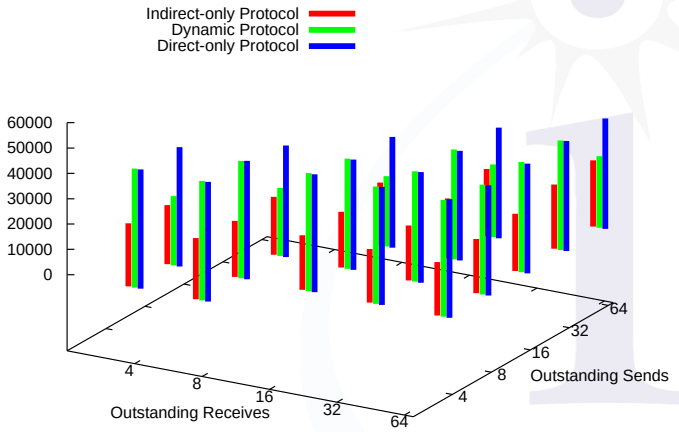
EXS throughput performance

- The bigger the message, the smaller the CPU usage (for fixed number of outstanding messages)
- The more simultaneously outstanding messages, the higher the throughput (for fixed message size)
- Reasonable “sweet spot”: 512 Kibibytes, 4 messages
  - throughput: 45.6 Gigabytes/second
  - CPU usage: 14.0% user, 9.4% kernel, 23.4% total
- Ideal “sweet spot”: 2 Mibibytes, 4 messages
  - throughput: 47.9 Gigabytes/second
  - CPU usage: 4.2% user, 2.3% kernel, 6.5% total

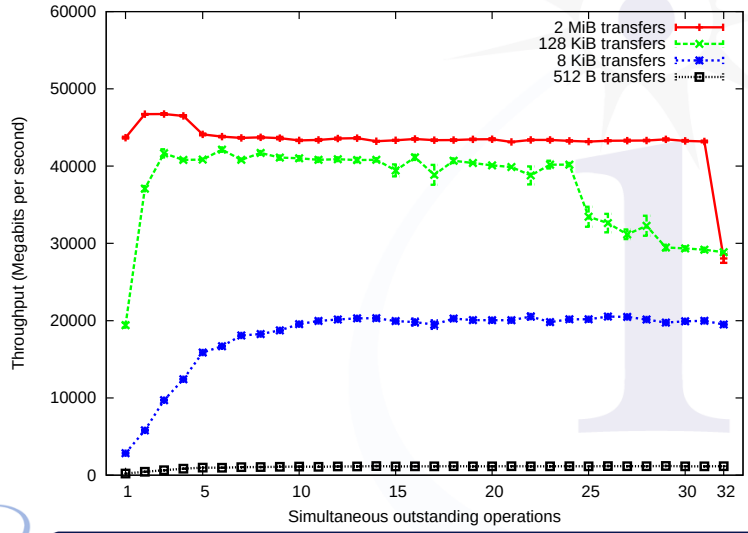
Patrick: EXS algorithm

- UNH EXS does zero-copy whenever possible
- For generic socket library, requires sender to wait for buffer advertisement
- Intermediate receive buffer allows sender to send immediately but is no longer zero-copy
- Solution: dynamic algorithm to choose direct or indirect transfers

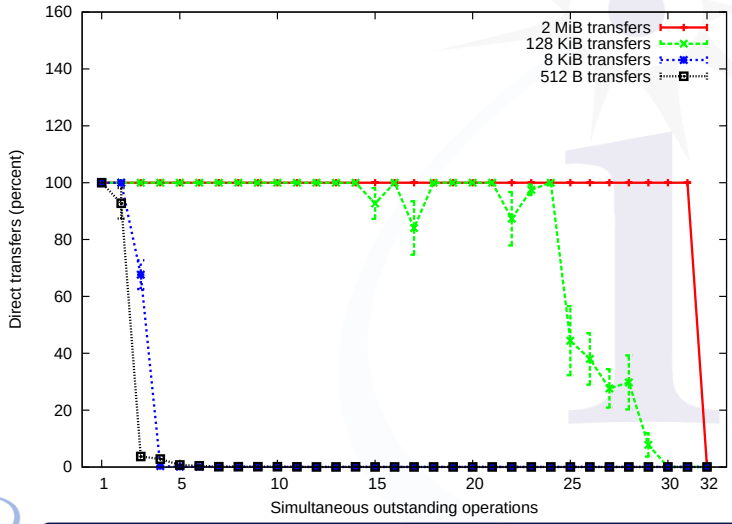
Throughput Comparison



Dynamic Algorithm Throughput



Dynamic Algorithm Percent Direct



MPI over RDMA run-time variance

- Measure and analyze time variation
  - Difficult in production CESM environment
  - Isolate network performance in IOL test environment
- UNH IOL test environment
  - Intel Westmere CPUs, 12 cores, 64 Gibibytes
  - InfiniBand QDR, PCIe-2 bus (ibdump cannot handle FDR yet)
  - InfiniBand SX6036 FDR-capable switch

MPI to RDMA Mapping

- To understand network traffic patterns from MPI calls
  - Data transfers
  - Collectives
  - Barriers
- Software tools used to gather data
  - EXTRAE – to get timing of MPI calls
  - PARAVER – to graphically display EXTRAE data
  - ibdump – to capture traces of RDMA (InfiniBand) traffic
  - Developed scripts to analyze data (UNH, UVA, NCAR)

Measuring MPI Performance over RDMA

- 9 different situations for send/recv
  - MPI operation: send, isend,ssend
  - MPI mode: eager, rendezvous-read, rendezvous-write
- Tools used to gather data
  - EXTRAE – to get timing of MPI calls
  - R – to obtain statistics from EXTRAE output
  - Scripts to massage data, plots
- Typical Output
  - 9 Box plots showing mean, 1st and 3rd quartiles, outliers for various MPI calls
  - 126 histograms total, consisting of 42 (all-points frequency, all-points value, outliers frequency) triplets

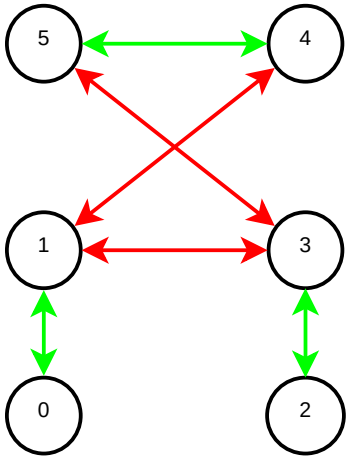
OpenMPI Eager vs. Rendezvous

- Eager protocol
  - Sends message immediately to MPI internal buffer, receiver must copy to user buffer
- Rendezvous protocol
  - Uses RDMA READ/WRITE operation to write directly to user buffer
    - Which operation is used is specified in openib_flags parameter
  - Used if:
    - message_size > openib_eager_limit (default 13KB)
    - messages received > openib_eager_rdma_threshold

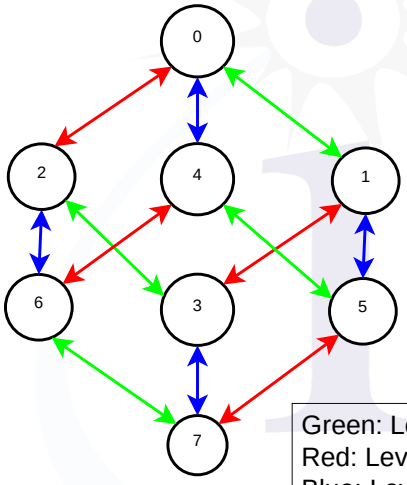
MPI_Send variants

- MPI_Send: simple blocking send
- MPI_Isend: nonblocking send
  - Caller must call MPI_Wait, MPI_Waitany, or MPI_Waitall to get completion event
- MPI_Ssend: blocking “synchronous” send
  - Does not return until receiver has actually received the message

MPI_AllReduce mapping



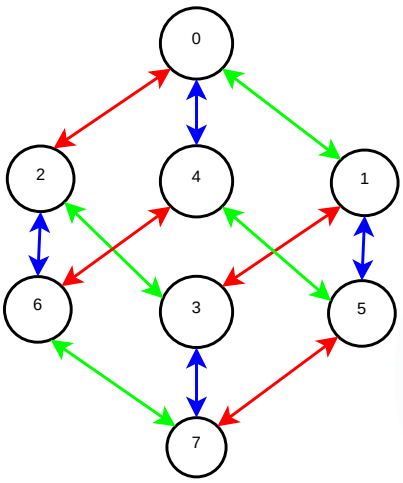
6 tasks



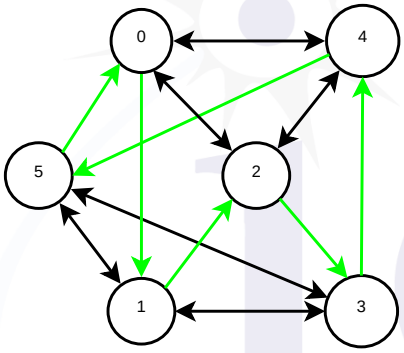
8 tasks

Green: Level 1
Red: Level 2
Blue: Level 3

MPI_Barrier mapping



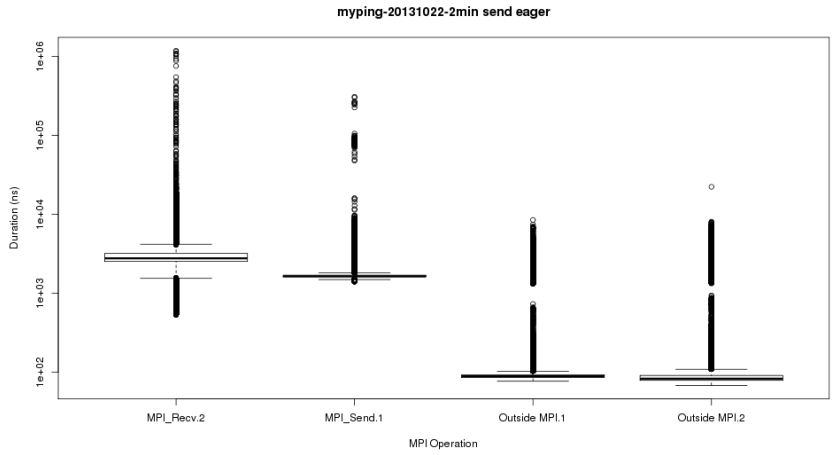
8 tasks

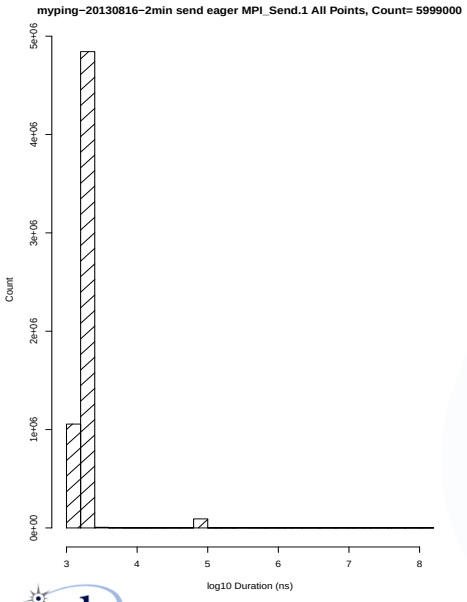


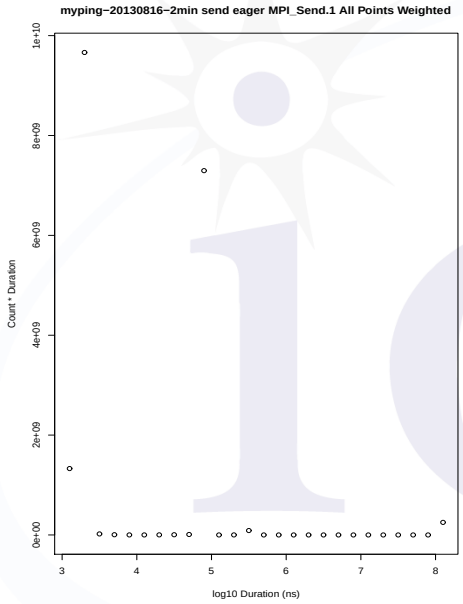
6 tasks

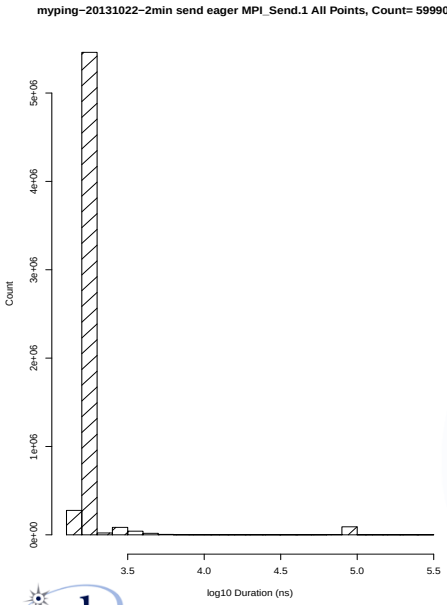
Green: Level 1
Red: Level 2
Blue: Level 3
Black: Level 2/3

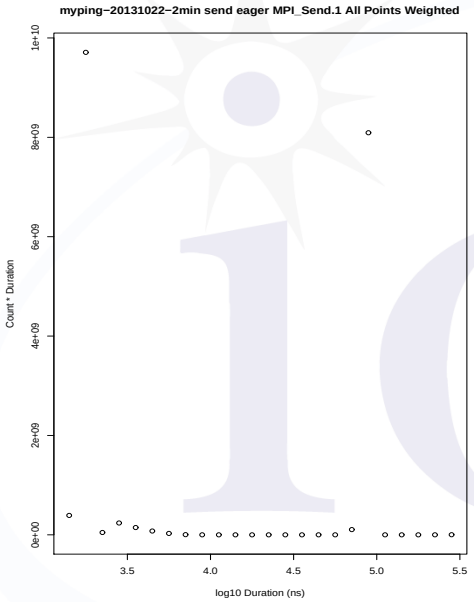
Patrick: example of outliers











Future work

- Develop simulation model of MPI/RDMA network traffic
  - Based on our MPI to RDMA Mappings and timings
  - Should model traffic congestion at switches
  - Should be able to reproduce observed behavior
  - Will enable us to study different network topologies and routing algorithms and their effect on performance
- Integrate this model with existing OMNet++ based Dimemas simulator for MPI programs
- Port VCMTP to RDMA as a reliable multicast
- Proposal to ANL for GridFTP modifications

THANK YOU!